Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 35 (2014) 358 – 367

Procedia Computer Science

18th International Conference on Knowledge-Based and Intelligent Information & Engineering Systems - KES2014

Parallel learning and classification for rules based on formal concepts

Nida Meddouri[a,*], Hela Khoufi[a], Mondher Maddouri[a,b]

[a]*Laboratoire d'Informatique, Programmation, Algorithmique et Heuristique - LIPAH*
Faculté des Sciences mathématiques, physiques et naturelles de Tunis - FST
Université d'El Manar
Campus universitaire El Manar, 1060, Tunis, Tunisie.
[b]*Department of Computer Sciences, Community College of Hinakya;*
Taibah University, Medina, Kingdom of Saudi Arabia.

Abstract

Supervised classification is a spot/task of *data mining* which consist on building a classifier from a set of instances labeled with their class (*learning step*) and then predicting the class of new instances with a classifier (*classification step*). In supervised classification, several approaches were proposed such as: *Induction of Decision Tree* and *Formal Concept Analysis*. The learning of formal concepts is generally based on the mathematical structure of *Galois lattice* (or *concept lattice*). The complexity of *Galois lattice* generation limits the application fields of these systems. In this paper, we discuss about supervised classification based on *Formal Concept Analysis* and we present methods based on *concept lattice* or *sub lattice*. We propose a new approach that builds only a part of the lattice, including the best concepts (i.e pertinent concepts). These concepts are used as classifiers in parallel combination using voting rule. The proposed method is based on *Dagging of Nominal Classifier*. Experimental results are given to prove the interest of the proposed method.

Peer-review under responsibility of KES International.

Keywords: Formal Concept, Ensemble method, Dagging, Classification Rules, Machine Learning, Data Mining.

1. Introduction

Formal Concept Analysis is a formalization of the philosophical notion of concept defined as a couple of extension and comprehension. The comprehension (called also intention) makes reference to the necessary and sufficient attributes which characterizes this concept. The extension is a set of instances which made it possible to find out the concept [1].

* Corresponding author. Tel.: +216-20-422-463.
E-mail address: nida.meddouri@gmail.com

Peer-review under responsibility of KES International.
doi:10.1016/j.procs.2014.08.116

The classification approach based on *Formal Concept Analysis* is a symbolic approach allowing the extraction of correlations, reasons and rules according to the concepts discovered from data. Supervised classification is a process made up of two steps. In the learning step, we organize the information extracted from a group of objects in the form of a lattice. In the classification step, we determine the class of new objects, based on the extracted concepts. Many learning methods based on *Formal Concept Analysis* were proposed, such as: GRAND[2], CLNN&CLNB[3], IPR[4], NAVIGALA[5], CITREC[6] and more recent BFC[7] and BNC[8]. Unfortunately, systems based on *Formal Concept Analysis* encountered some problems such as exponential complexity (in the worth case), high error rate and over-fitting[8,9].

In this last decade, a great number of researches in machine learning have been concerned with the ensemble methods of classifiers that allow the improvement of a single learner performance (generally a weak learner) by the voting techniques[10]. In the area of supervised learning, several ensemble methods have been appeared[11] such as *Boosting* and *Bagging* which improve the performance of combined classifiers sets. The two principal reasons for this success are probably the simplicity of implementation and the recent theorems relative to the boundaries, the margins, or to the convergence[10,12,13]. Generally, the ensemble methods are based on sequential or parallel learning (*Bagging*). The difference between them derives from how to select data for learning.

In sequential learning such as *Boosting*, all the data are considered in each learning step and the weights are assigned to learning instances. However, it was proved that this method is not interesting and no sufficient for a more efficient classifier as *Decision Tree*[8]. In parallel learning, such as *Bagging*, the training data are drawn randomly with replacement from the original data set, such a training set is called a *Bootstrap*. The well known method which is based on parallel learning is *Dagging* (***D****isjoint samples* ***agg****regat****ing***), it creates a number of disjoint groups and stratified data from the original learning data set, each one is considered as a subset of learning. The weak learner is built on this learning sets. The predictions are then obtained by combining the classifier outputs by majority voting[14]. *Dagging* has shown its importance in recent work. Then, we propose to use this technique, in this work, to study the classifier ensembles based on formal concepts, since, no study has focused on the formal concepts in the context of parallel learning.

In section 2, we present a state of the art on *Formal Concept Analysis* and several methods used which are based on lattice concept and sub-lattice of concepts. In section 3, we propose a new method exploiting the advantages of the *Dagging* to generate and combine in parallel way weak concept learners[15,16]. From the section 4, a comparative experimental study is presented to evaluate the performance of parallel classifier ensembles according to certain criteria such as the number, variety and the type of classifiers. A comparative experiment is also presented to show the importance of parallel learning using stratified sampling, compared to sequential learning using random sampling.

2. Formal concept analysis and classification

2.1. Definition

A formal context is a triplet $< O, \mathcal{P}, \mathcal{R} >$, where $O = \{o_1, o_2, ..., o_n\}$ is a finite set of n instances, $\mathcal{P} = \{p_1, p_2, ..., p_m\}$ a finite set of m properties (binary attributes) and $\mathcal{R}$ is a binary relation defined between O and $\mathcal{P}$. The notation $(o_i, p_j) \in \mathcal{R}$ or $\mathcal{R}(o_i, p_j) = 1$ means that the instance o_i verifies the property p_j in relation $\mathcal{R}$[1]. The context (see Table 1 and 2)[1] is often represented by a cross-table or a binary-table.

Let $A \subseteq O$ and $B \subseteq \mathcal{P}$ be two finite sets. For both sets A and B, operators $\varphi(A)$ and $\delta(B)$ are defined as[1]:

- $\varphi(A) = \{p \mid \forall o,\ o \in A \text{ and } (o, p) \in \mathcal{R}\}$.
- $\delta(B) = \{o \mid \forall p,\ p \in B \text{ and } (o, p) \in \mathcal{R}\}$.

Operator φ defines the properties shared by all elements of A. Operator δ defines instances which share the same properties included in set B. Operators φ and δ define a Galois connexion between sets O and $\mathcal{P}$[1]. The closure operators are $A" = \varphi \circ \delta(A)$ and $B" = \delta \circ \varphi(B)$. Finally, the closed sets A and B are defined by $A = \varphi \circ \delta(A)$ and $B = \delta \circ \varphi(B)$.

[1] The data sets is selected from UCI Machine Learning Repository[17]

Table 1. Illustration of the formal context (data Weather under binary format).

O-$\mathcal{P}$	p_1	p_2	p_3	p_4	p_5	p_6	p_7	p_8	*CLASS*
o_1	1	0	0	1	0	0	0	1	***2***
o_2	1	0	0	1	0	0	0	0	***2***
o_3	0	1	0	1	0	0	0	1	***1***
o_4	0	0	1	0	1	0	0	1	***1***
o_5	0	0	1	0	0	1	1	1	***1***
o_6	0	0	1	0	0	1	1	0	***2***
o_7	0	1	0	0	0	1	1	0	***1***
o_8	1	0	0	0	1	0	0	1	***2***
o_9	1	0	0	0	0	1	1	1	***1***
o_{10}	0	0	1	0	1	0	1	1	***1***
o_{11}	1	0	0	0	1	0	1	0	***1***
o_{12}	0	1	0	0	1	0	0	0	***1***
o_{13}	0	1	0	1	0	0	1	1	***1***
o_{14}	0	0	1	0	1	0	0	0	***2***

Table 2. Specification of attributes

Attributes	*Signification*
p_1	*Outlook=Sunny*
p_2	*Outlook=Overcast*
p_3	*Outlook=Rainy*
p_4	*Temperature=Hot*
p_5	*Temperature=Mild*
p_6	*Temperature=Cool*
p_7	*Humidity*
p_8	*Windy*

A formal concept of the context $< O, \mathcal{P}, \mathcal{R} >$ is a pair (A, B), where $A \subseteq O$, $B \subseteq \mathcal{P}$, $\varphi(A) = B$ and $\delta(B) = A$. Sets A and B are called, respectively, the *extent (domain)* and the *intent (co-domain)*.

From a formal context $< O, \mathcal{P}, \mathcal{R} >$, we can extract all possible concepts. The set of all concepts may be organized as a complete lattice (called *Galois lattice*[1]), when the following partial order relation '$\ll$' is defined between two concepts $(A_1, B_1) \ll (A_2, B_2)$ if and only if $(A_1 \subseteq A_2)$ and $(B_2 \subseteq B_1)$. The concepts (A_1, B_1) and (A_2, B_2) are called nodes in the lattice.

2.2. Classification

The classification has to determine the class of new objects. The *Galois lattice* can be seen as a space of search in which we evolve level to another, by validating the characteristics associated to the concepts[9]. In the literature, many existing classification systems are based on complete lattice or sub lattice of concepts.

Complete lattice based methods: They use lattice concept such as `GRAND`[2] and `NAVIGALA`[5]. There are three common limits for systems based on concept lattice. First, the complexity of the lattice generation (temporally and spatially) is exponential. Then, the navigation in huge search space is hard[18]. In addition, the data used is binary. For these reasons, many researchers are focused on the sub-lattice based classification.

Sub-lattice based methods: There are methods which have the characteristic to build sub-lattice which reduces their theoretical complexity and their execution times. A sub-lattice is a reflexive and transitive reduction of `Galois` lattice. Classification based on sub-lattice is similar to that started from a lattice. The major difference between lattice based classification and sub-lattice based classification is the number of concepts generated. Systems like `CLNN&CLNB`[3], `IPR`[4], `BFC`[7] and `BNC`[8], have the characteristic of building a part of the concept lattice and inducing classification rules. Except `IPR`, `BFC` and `BNC`, the other systems extract the classification rules from the sub higher lattice. The methods based on sub-lattice of concepts generate less classification rules than one based on a complete lattice of concepts. However, their limit is the possible loss of information in a condensed data representation or in a partial reproduction of the complete lattice. We remark that with the methods based on sub-lattice classification, the constructed concepts are chosen based on inappropriate criteria (i.e. the depth of the lattice, the covering of the context, etc.)[9].

Now, if we take all the supervised learning methods based on *Formal Concept Analysis*, we can report their following common limits: absence of the adaptive aspect, handling only binary data and a high complexity. Also, the construction of the concepts is exhaustive or non-contextual. That is why recently, some researches in machine learning have converted to the integration and the use of ensemble learning for *Formal Concept Analysis* that allow to improve the single learner performance by voting techniques[10].

In[7] and[8], the authors presented the `BFC` (***B**oosting of **F**ormal **C**oncepts*) method based on *Formal Concept Analysis* and exploiting the advantages of *Boosting* algorithm. This method handles with binary data only and uses *AdaBoost.M2* which is the basic algorithm of multi class *Boosting*[19]. Initially, the algorithm attributes equal weights

to the learning instances. It selects a part of the learning data and extracts the pertinent concept within the binary data sets. It uses the discovered formal concept to classify the learning data and updates the weights of the learning instances by decreasing those of the well classified ones and increasing the weights of the others (the bad instances). After that, it repeats the resampling based on the new weights, in order to discard the well classified instances and to consider only the bad ones. The BFC method builds adaptively a part of the concept lattice made up only by pertinent formal concepts. The method of BFC has the particularity to decide the number of iterations. So that, it can control the time of execution and gives the best decision in all iterations. BNC (***B**oosting of **N**ominal **C**oncepts*) is an extension of BFC that handles with nominal data and builds adaptively pertinent *nominal concepts*[8].

According to[13], the adaptive update of learning data in sequential learning (*Boosting*) increases the misclassified weight by the previous classifier and improves the performance of any learning algorithm (*weak learner*). However, the capacity of sequential learning have been challenged when highly noisy data are used. In the case of parallel learning, the noisy data is ignored and possibly will spread equiprobably between *Bootstraps* or other re-sampled subsets of training data. That is why techniques such as *Bootstrapping*[10] and *Disjoints Stratified*[20] have been proposed ignoring these noisy data or to distribute on different sets of learning[21]. In the next section, we propose to take advantage of parallel learning and to generate *nominal classifiers* from *disjoints stratified* data.

3. Dagging of nominal concepts

In parallel approach, the generation of classifiers is based on *Bootstraps* (in *Bagging*) or from other resampled data sets. The peculiarity of these training sets is to reduce the impact of hard to learn instances (called *outliers* and *misleaders*)[21]. In *Bootstrapping*, n' instances are selected and drawed randomly with replacement from the original training set of n training instances with $n' <= n$. Each classifier is then trained on this set, such a training set is called a *Bootstrap* replicate of the original set. Each *Bootstrap* replicate contains, on average, *63.2%* of the original training set, with many instances appearing multiple times. Predictions on new instances are made by taking the majority vote of the ensemble.

In the literature, stratified sampling has proved to be efficient[14]. Learning from stratified data samples allows to generate more efficient classifier than those generated from the weighted data in the case of sequential learning classifiers. *Dagging* has the particularity to learn classifiers in parallel way from stratified data sets. We propose to exploit this varient of parallel learning method to generate classifiers based on *Nominal Concepts*.

3.1. Classifier based on nominal concepts (CNC)

Input: Sequence of n instances $O = \{(o_1, y_1), ..., (o_n, y_n)\}$ with labels $y_i \in \mathcal{Y} = \{1, ..., Y\}$.
Output: The classifier rule $h_{\mathtt{CNC}}$.
begin
From O, find the attribute having the best *Informational Gain* value AN^* ;
From AN^*, find the nominal value having the important efficient v ;
Calculate the closure associated to v $(\{\delta(AN^* = v)\}, \delta \circ \varphi(\{AN^* = v\}))$;
Determine the majority class y^* associated to $\delta(AN^* = v)$;
Induce the classification rule $h_{\mathtt{CNC}}$;
Return $h_{\mathtt{CNC}}$;
end

Algorithm 1: Algorithm of **C**lassifier **N**ominal **C**oncept (CNC)

The learning algorithm CNC (Algorithm 1) consider the whole of nominal training instances O described by L nominal attributes AN (which are not necessary binary).

$$AN = \{AN_l \mid l = \{1, .., L\},\ \exists o_i \in O,\ \exists p \in \mathcal{P},\ AN_l(o_i) = p\}. \quad (1)$$

The pertinent nominal concept is extracted within the data set by selecting the nominal attribute which minimises the measure of *Informational Gain* (*IG*). *CNC* calculates the *IG* of each attribute from the learning context of *n* instances with:

$$IG(AN, O) = E(O) - \sum_{j=1}^{Val.Att} \frac{S(v_j)}{n} E(v_j) \quad (2)$$

IG of the nominal attribut *AN* (represented by *Val.Att* different values) is calculated from the *Entropy* function (*E()*). *S()* calculates the relevance of a value v_j of the attribut *AN* on the whole O. The variation of *IG* depends on $S(v_j)$ if we neglect the variation $E(O)$. The data set that contains redundant instances (as simple random samples) maximizes the value of $S(v_j)$ (if an instance containing v_j is redundant) and minimizes the *IG* of the corresponding attribute. This paralyzed effect does not bring a lot of diversity in the values of *IG* for the same attribute. In a diverse set of data (such as stratified samples), value of $S(v_j)$ is minimized and the *IG* of the corresponding attribute is more important. This phenomenon helps to better enhance the attributes of a diverse set. The stratified random sampling ensures the proportional presence of all the various sub-groups within the data set. Clearly, finding the best attribute that maximizes the *IG* in a stratified set is more interesting than in other set. So, we recommend to learn `CNC` from stratified sets and based on the calculation of *IG*.

Once the nominal attribute is selected (AN^*), we extract the associated instances for each value v_j from the selected attribute according to the proposition 1.

Proposition 1: From a nominal context (multi-valued), the δ operator is set by (3):

$$\delta(AN^* = v_j) = \{o \in O \mid AN^*(o) = v_j\}. \quad (3)$$

Then, we look for the other attributes describing all the extracted instances (using the closure operator $\delta \circ \varphi(AN^* = v_j)$). For this, we give the following proposition.

Proposition 2: From a nominal context (multi-valued), the φ operator is set by (4):

$$\varphi(B) = \{v_j \mid \forall\, o, o \in B \text{ and } \exists\, AN_l \in AN \mid AN_l(o) = v_j\}. \quad (4)$$

So, we construct the pertinent concept associated to each value v_j of the best attribute AN^* ($\delta(AN^* = v_j), \delta \circ \varphi(AN^* = v_j)$). A weak classifier is obtained by seeking the majority class associated to the extent of the pertinent concept ($\delta(AN^* = v_j)$). It induces a classification rule. The condition part of the rule is made up by the conjunction of the attributes included in the intent: $\delta \circ \varphi(AN^* = v_j)$. The conclusion part of the rule is made up by the majority class. After that, we use the discovered rule to classify the learning data set O and so our proposed learning algorithm of pertinent concept stops at this iteration.

We study the standard deviations of the error rate of `CNC` on 15 samples of different data[2]. The performance of `CNC` is obtained according to the 10 cross-validation method. We report the standard deviations of the error rate for each data set (Table 3). These standard deviations are more or less important, showing that `CNC` is an unstable classifier.

3.2. Learning concept based classifiers

The proposed approach is essentially based on *Dagging* described with more details in Algorithm 2 [15,16]. To generate T classifiers, we execute T times the learning algorithm on various disjoint and stratified sets of learning instances. Each set of learning instances is satisfied to have a similar distribution to the initial set. The samples are obtained by drawing n' instances randomly without replacement in the training sample O, with $n' < n$. These samples respect the distribution of learning instances as classes.

The principle of `DNC` (***D**agging **N**ominal **C**lassifier*) is then to take several disjoint and stratified samples $\{O^{\Theta_1}, ..., O^{\Theta_T}\}$. On each of which, the `CNC` is built to get a collection of classifiers $\{h_1, ..., h_T\}$ and to combine them by majority voting rule [14].

[2] Presented with more details in section 4

Table 3. Performance of individual CNC.

Data Sets	*Error Rates*	*Standard Deviations*
Car	7.47%	8.01%
Kr-vs-kp	33.95%	1.80%
Waveform	13.18%	1.30%
Optdigits	28.36%	1.55%
Nursery	12.67%	4.47%
Pendigits	9.68%	0.84%
German credit	4.60%	1.51%
Japanese vowels	18.45%	1.56%
Splice	33.10%	2.24%
Segment	7.01%	1.10%
Spambase	6.56%	0.69%
Cmc	34.49%	2.58%
Solar-flare	0.19%	0.39%
Page-blocks	1.17%	0.45%
Yeast	40.84%	3.08%

Input:
1. T: number of classifiers to generate.
2. Learning data O of n instances with $O = \{(o_1, y_1), ..., (o_n, y_n)\}$ labeled $y_i \in \mathcal{Y} = \{1, ..., Y\}$.

Output: The final classifier h_{vote}.
begin
  Divide the population into S strates;
  Establish the most complete list of each S constituting strates;
  for *t from 1 to T* **do**
    Calculate the percentage P_t instances of each strate with respect to O ;
    Choose simply and randomly instances from each strate to form O^{Θ_t} respecting P_t;
    Learn CNC on O^{Θ_t} to generate h_t;
  end
  $h_{vote} = arg\ max_{y \in \mathcal{Y}} \sum_{t=1}^{T} h_t(o, y)$;
end

Algorithm 2: Algorithm of **D**agging **N**ominal **C**lassifier (DNC)

4. Experimental study

In this section, we study the behaviour of proposed method called DNC to see in which conditions it can improve the performance of individual weak CNC. In particular, we will try to provide answers to the following questions: Does the number of classifiers have an effect on the performance of *Dagging* of CNC? Is *Dagging* of CNC more interesting than other classifiers? What is the best adaptive learning for CNC: sequentially or parallel? What are the conditions under which CNC behaves better than other classifiers?

We compare the DNC method with existing classifiers in the literature as Bayes Net, Id3, J48, Decision Stumps and CITREC[6] (based on *Formal Concept Analysis*) on some well data sets extracted from "UCI Machine Learning Repository"[17].

The performance of generated classifiers is evaluated in terms of error rate. To calculate these rates, the *10 cross-validation* method is used in WEKA whose principle is to divide each base on 10 subsets. In turn, each subset is used for testing and the other subsets for learning.

Table 4. Characteristics of data sets used.

Data Sets	*Instances*	*Attributes*		*Classes*	*Data*
		Numeric	*Nominal*		*Diversity*
Car	1728	6	6	4	100%
Kr vs kp	3196	36	36	2	100%
Waveform	5000	40	40	3	100%
Optdigits	5620	64	64	10	100%
Nursery	12960	8	8	5	100%
Pendigits	10992	16	16	10	99.18%
Credit German	1000	20	20	2	98.59%
Japanese Vowels	5687	12	14	9	97.06%
Splice	3190	61	60	3	94.42%
Segment	2310	19	19	7	84.97%
Spambase	4601	57	57	2	78.26%
CMC	1473	9	9	3	64.96%
Solar flare	1066	10	12	6	34.30%
Page-blocks	5473	10	10	5	23.14%
Yeast	1484	8	8	10	22.34%

The chosen data sets were discretized with 2 discretional filters under `WEKA`[3]. The first filter[4] is an instance filter that converts a range of numeric attributes into nominal attributes (to evaluate `Bayes net`, `Id3`, `J48`, `Decision Stumps` and `CNC`). The second filter[5] is an instance filter that converts a range of nominal attributes into binary attributes (to evaluate methods based on *Formal Concept Analysis*). These data sets are presented in Table 4. For each data sample, we present respectively the number of instances, the number of numeric attributes (before discretization), the number of nominal attributes (after discretization) and the number of classes. The last column presents the level of diversity of data to see if it affects the performance of the *Dagging*. This diversity of data is the ratio between the number of different vectors of instances (attributes) and the total number of vectors in each database[22].

4.1. Influence of the classifier numbers

Table 5. Dagging performance when increasing the classifier numbers.

Data Sets	2	3	4	5	6	7	8	9	10	11	12	13
Car	9,04%	7,41%	14,35%	8,9%	12,67%	10,3%	8,1%	**6,3**%	11,11%	6,31%	11,34%	7,69%
Kr-vs-kp	33,95%	33,95%	33,95%	33,95%	33,85%	33,85%	33,92%	33,92%	33,95%	33,95%	32,89%	**32,88**%
Waveform	13,18%	12,7%	12,58%	12,56%	12,74%	12,3%	12,14%	12,86%	13,32%	**11,44**%	12,02%	11,52%
Optdigits	28,35%	28,29%	28,26%	28,31%	28,35%	28,22%	33,08%	28,42%	28,86%	**27,38**%	28,26%	28,47%
Nursery	13,06%	14,35%	14,46%	12,89%	14,73%	14,41%	14,58%	14,21%	14,79%	**11,85**%	14,47%	14,77%
Pendigits	9,66%	9,68%	9,86%	11,33%	11,69%	11,71%	**9,55**%	11,57%	16,74%	9,66%	13,18%	11,83%
German credit	**4,6**%	6,4%	7,3%	7,4%	8,3%	8,8%	7,3%	9,7%	11%	10,4%	8,7%	9,2%
Japanese vowels	18,45%	16,72%	17,55%	16,6%	15,4%	15,53%	14,24%	13,84%	17,71%	17,44%	17,5%	**12,76**%
Splice	33,1%	33,1%	32,82%	32,13%	33,1%	31,32%	**28,4**%	31,57%	29,5%	33,1%	29,97%	28,53%
Segment	7,01%	6,97%	7,01%	6,36%	6,23%	6,32%	5,93%	5,5%	**4,37**%	6,54%	7,1%	5,58%
Spambase	7,35%	7,3%	7,28%	8,26%	8,46%	11,13%	8,22%	8,85%	7,28%	9,37%	**6,52**%	11,13%
Cmc	34,02%	33,34%	32,65%	29,53%	31,44%	30,62%	32,12%	30,01%	**29,33**%	29,6%	31,44%	31,77%
Solar-flare	0,37%	0,09%	0,09%	0,09%	0,19%	0,19%	0%	0,09%	**0**%	0,09%	**0**%	**0**%
Page-blocks	1,17%	1,17%	1,17%	1,17%	1,17%	1,3%	**1,13**%	1,17%	**1,13**%	1,17%	1,15%	**1,13**%
Yeast	40,84%	40,84%	40,84%	40,77%	40,97%	40,64%	40,44%	40,57%	40,37%	**39,69**%	40,84%	41,38%

To study the performance of *Dagging* using `CNC`, we generated sets by varying the classifier number from 2 to 13 and we reported their error rates in Table 5. From this table, we can first report that the performance of odd sets of classifiers is better than the performance of pair sets. This is due to the combination rule that works better with an odd number of classifiers[23]. With more than 8 classifiers, *Dagging* behaves better. These ensembles are not correlated

[3] Available at `http://www.cs.waikato.ac.nz/ml/Weka`

[4] `weka.filters.supervised.attribute.Discretize`

[5] `weka.filters.supervised.attribute.NominalToBinary`

with the training data diversity. We note also that with 11 classifiers, *Dagging* produces the best performance. The average errors (for all the bases) occurred with 11 classifiers (16.53%) is lower than those obtained with 13 classifiers (16.58%). That is why in the next experiments, we will retain this number (T=11) in the generation of classifier ensembles.

4.2. Influence of the classifier type

Table 6. Dagging performance using different classification methods.

Data Sets	Bayes Net	CNC	Id3	J48	Decision Stumps	CITREC
Car	17,13% (1.95)	**6,31**% (8.21)	11,51% (2.67)	18,69% (3.13)	29,98% (0.17)	23,79% (2.34)
Kr-vs-kp	12,52% (1.89)	33,95% (1.8)	**1,91**% (0.67)	3,1% (1.02)	33,95% (1.8)	47,78% (0.1)
Waveform	19,32% (1.43)	**11,44**% (2.57)	19,9% (2.11)	22,12% (1.89)	48,5% (2.36)	41,42% (6.89)
Optdigits	**7,92**% (1.01)	27,38% (3.9)	21,12% (0.96)	17,62% (1.63)	56,44% (4.76)	- (-)
Nursery	9,82% (0.85)	11,85% (6.31)	**4,88**% (0.52)	8,56% (0.57)	33,75% (0.04)	- (-)
Pendigits	12,85% (1.03)	**9,66**% (0.87)	12,43% (0.95)	16,48% (1.25)	74,75% (3.47)	- (-)
German credit	23,6% (3.92)	**10,4**% (7.5)	27,9% (4.25)	28,6% (2.63)	30% (0.94)	30% (0)
Japanese vowels	48,51% (1.65)	**17,44**% (7.22)	44,59% (1.72)	45,44% (2.1)	59,13% (0.11)	67,49% (2.14)
Splice	**5,11**% (1.33)	33,1% (2.24)	8,31% (1.69)	11,5% (2.47)	27,21% (8.89)	- (-)
Segment	10,48% (1.47)	**6,54**% (2.71)	8,14% (2.25)	9,35% (1.45)	70,22% (3.83)	- (-)
Spambase	9,87% (0.95)	9,37% (4.75)	**8,93**% (1.3)	10,3% (1.57)	15,54% (3.8)	- (-)
Cmc	47,66% (5.29)	**29,6**% (4.99)	45,56% (4.54)	46,23% (3.11)	54,79% (1.94)	55,33% (2.05)
Solar-flare	1,88% (1.17)	**0,09**% (0.3)	0,47% (0.49)	0,47% (0.49)	0,47% (0.49)	- (-)
Page-blocks	6,47% (0.89)	**1,17**% (0.45)	3,51% (0.65)	5,32% (0.57)	6,8% (0.43)	- (-)
Yeast	41,18% (5.22)	**39,69**% (3.39)	42,46% (3.07)	42,32% (4.66)	59,3% (1.18)	- (-)

In[8], we found that the sequential learning is beneficial for classifiers such as J48 and Id3. However, the CNC is among the worst classifiers. Our objective here is to study the behavior of these classifiers in the case of parallel learning and to see whether the CNC fits better with this technique. To do that, for each type (ID3, Bayes Net, CNC, J48 and CITREC), ensembles of 11 classifiers are generated in *Dagging*. The error rates of these sets and the *Standard Deviations* are reported in Table 6. Signs '-' indicates that the method can not process the sample data, due to excessive consumption of memory resources. From these results, *Dagging* of CNC based on the *Formal Concept Analysis* holds the best performance for 10 data sets from 15. Id3 and Bayes Net are better than J48. Decision Stumps produced the higher error rates compared with the rest of the classifiers.

For sets of correlated data such as *Pages blocks* and *Solar Flare* having diversity values of 23.14% and 34.3%, respectively, the error rates of nominal classifiers (CNC) are lower. For the *Yeast* data set having 22.34% as diversity value, the error rates are quite higher. This shows that the data diversity is not correlated with the performance of nominal classifier ensembles.

Dagging of nominal classifiers (DNC) is sensitive to the number of attributes. It produces excellent results for data sets with a reasonable number of attributes (not exceeding 40 attributes), whatever the size of the data sets. This allows us to deduce that DNC will be very interesting for classification problems whose bases are very large. From these experiments, we can note that parallel learning is more interesting for our nominal classifier CNC. The Other classification methods can be rather used in sequential learning (as shown in[8]).

4.3. Comparison of ensemble methods

According to the literature, the relationship between the type of classifier and the ensemble methods is not clear: for example we do not know whether it is better to use *Boosting*, *Bagging* or *Dagging* for a given classification problem. In[14], the authors have shown, theoretically and experimentally, the importance and reliability of *Dagging*.

In[24], we noticed that CNC is not a good enough using sequential learning on different sizes of data sets. To see which is better for our CNC, the parallel or sequential learning, we generate sets of 11 nominal classifiers by *Boosting*, *Bagging* and *Dagging* on 15 different data sets (Table 4). Figures 1 and 2 presents the results of all these methods, in terms of error rate and training time. These results show that *Dagging* is the best ensemble method producing low error rates for all generated classifier sets. All the errors rates of *Boosting* are higher than those of *Dagging* (Figure

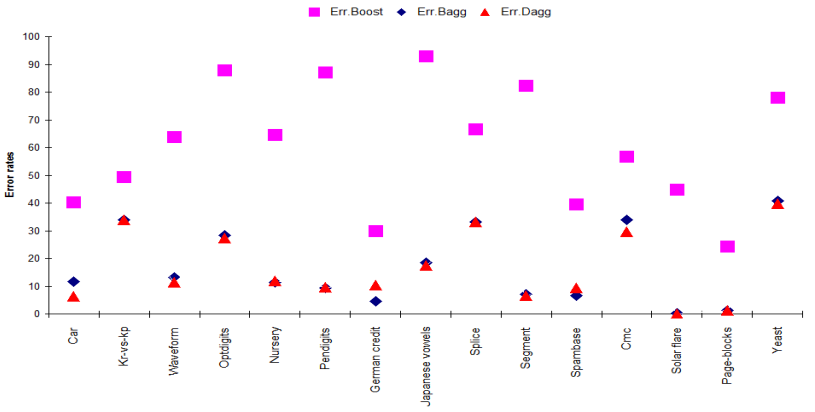

Fig. 1. Error rates of *Boosting*, *Bagging* and *Dagging* using `CNC`

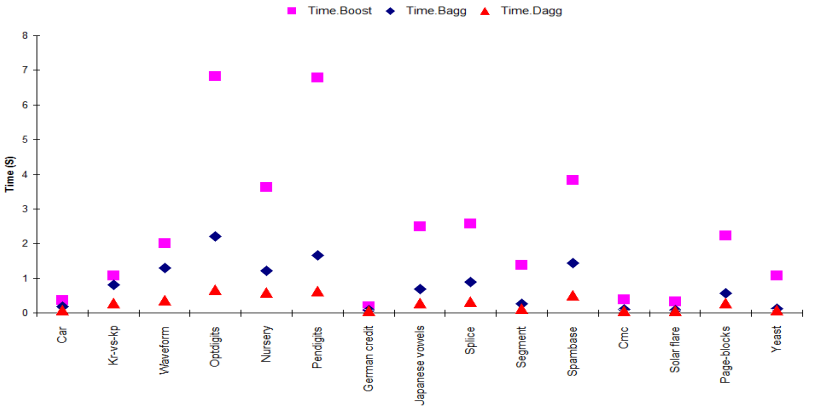

Fig. 2. Training time of *Boosting*, *Bagging* and *Dagging* using `CNC`

1). In addition, *Boosting* is 6 times slower than *Dagging*. We report also that *Dagging* is 2 times faster than *Bagging* (Figure 2). All ensemble methods does not depend on the diversity of data, because the performance is different regardless the level of diversity (no correlation).

In our experiments, learning ensembles of 11 nominal classifiers by *Dagging* is more interesting than by *Boosting*. It still similar to *Bagging* and more than a little for classification problems with 2 class. The parallel learning for classifiers based on *Formal Concept Analysis* is then better than the sequential learning especially for classification problems with several classes.

5. Conclusion

In this paper, we focused on the parallel learning of nominal concept classifiers. We proposed a new classifier based on nominal concepts that is better than the methods based on *Formal Concept Analysis*. We propose next to improve its performance by using ensemble methods because on the one hand the recent works have encouraged their use for linear classifiers and on the other, there is no study on parallel learning for *formal concept* classifiers. Particularly, we propose a new variant of *Dagging* to generate and combine ensembles of proposed classifier that seems to be a weak classifier.

We recommend a parallel learning by *Dagging* for classifiers such *formal concept* and the sequential learning for the other type of classifiers. In parallel learning, a few classifiers are sufficient for obtained better performance than the use of individual one. More experiments are possible on larger data sets with other ensemble methods, such as *Random Forests*, and for other classifiers as `J48` and `Id3`.

Many improvements on the ensemble methods can be brought. `BNC` and `DNC` methods used majority vote, for classifier combination. A variety of voting rules already exists. A study of these rules can be beneficial to improve the performance of `CNC` ensembles. Concerning the `CNC` algorithm, other measures can be used to select the best attribute because the concept of measures committee has shown its evidence in recent research[25]. We can study the combination of measures to adopt an appropriate committee to our classifier.

References

1. G. S. B. Ganter, R. Wille, Formal Concept Analysis: Foundations and Applications, Springer, 2005.
2. G. D. Oosthuizen, The use of a lattice in knowledge processing, Ph.D. thesis, University of Strathclyde, Glasgow, Scotland, UK (1988).
3. Z. Xie, W. Hsu, Z. Liu, M. L. Lee, Concept lattice based composite classifiers for high predictability, Journal of Experimental and Theoretical Artificial Intelligence 14 (2-3) (2002) 143–156.
4. M. Maddouri, Towards a machine learning approach based on incremental concept formation, Journal of Intelligent Data Analysis 8 (3) (2004) 267–280.
5. M. Visani, K. Bertet, J.-M. Ogier, Navigala: an original symbol classifier based on navigation through a galois lattice, International Journal of Pattern Recognition and Artificial Intelligence 25 (04) (2011) 449–473.
6. B. Douar, C. C. Latiri, Y. Slimani, Approche hybride de classification supervisée à base de treillis de galois: application à la reconnaissance de visages, in: Actes des 8mes Journes Francophones en Extraction et Gestion des Connaissances, Vol. E-11 of Revue des Nouvelles Technologies de l'Information, Cépaduès-Éditions, 2008, pp. 309–320.
7. N. Meddouri, M. Maddouri, Boosting formal concepts to discover classification rules, in: Proceeding of the 22rd International Conference on Industrial, Engineering & Other Applications of Applied Intelligent Systems, Vol. 5579 of Lecture Notes in Computer Science, Springer, 2009, pp. 501–510.
8. N. Meddouri, M. Maddouri, Adaptive learning of nominal concepts for supervised classification, in: Proceedings of the 14th International Conference on Knowledge-Based and Intelligent Information and Engineering Systems, Vol. 6276 of Lecture Notes in Computer Science, Springer, 2010, pp. 121–130.
9. N. Meddouri, M. Maddouri, Classification methods based on formal concept analysis, in: Proceedings of the 6th International Conference on Concept Lattices and Their Applications, 2009, pp. 9–16.
10. L. Breiman, Bagging predictors, Journal of Machine Learning 24 (2) (1996) 123–140.
11. L. I. Kuncheva, Combining Pattern Classifiers: Methods and Algorithms, Wiley-Interscience, 2004.
12. L. Kuncheva, M. Skurichina, R. P. W. Duin, An experimental study on diversity for bagging and boosting with linear classifiers, Journal of Information Fusion 3 (4) (2002) 245–258.
13. M. K. Warmuth, K. A. Glocer, G. Rätsch, Boosting algorithms for maximizing the soft margin, in: Proceedings of the Twenty-First Annual Conference on Neural Information Processing Systems, Curran Associates, Inc., 2007, pp. 1585–1592.
14. K. M. Ting, I. H. Witten, Stacking bagged and dagged models, in: In Proc. 14th International Conference on Machine Learning, 1997, pp. 367–375.
15. S. K. B., D. Kanellopoulos, Combining bagging, boosting and dagging for classification problems, in: Proceedings of the 11th International Conference on Knowledge-Based Intelligent Information and Engineering Systems, Vol. 4693 of Lecture Notes in Computer Science, Springer, 2007, pp. 493–500.
16. S. K. D. Anyfantis, M. Karagiannopoulos, P. Pintelas, Local dagging of decision stumps for regression and classification problems, in: Proceedings of 15th IEEE Mediterranean Conference on Control and Automation, 2007, pp. 1–6.
17. A. Asuncion, D. J. Newman, UCI machine learning repository (2007).
18. H. Fu, H. Fu, P. Njiwoua, E. M. Nguifo, A comparative study of fca-based supervised classification algorithms, in: Proceeding of Second International Conference on Formal Concept Analysis, 2004, pp. 313–320.
19. Y. Freund, R. E. Schapire, Experiments with a new boosting algorithm, in: Proceedings of the Thirteenth International Conference on Machine Learning, Morgan Kaufmann, 1996, pp. 148–156.
20. K. M. Ting, I. H. Witten, Stacking bagged and dagged models, in: In Proc. 14th International Conference on Machine Learning, Morgan Kaufmann, 1997, pp. 367–375.
21. M. Skurichina, R. P. W. Duin, Bagging for linear classifiers, Journal of Pattern Recognition 31 (7) (1998) 909 – 930.
22. M. S. Haghighi, A. Vahedian, H. S. Yazdi, Creating and measuring diversity in multiple classifier systems using support vector data description, Applied Soft Computing 11 (8) (2011) 4931–4942.
23. L. I. Kuncheva, C. J. Whitaker, R. P. W. Duin, Limits on the majority vote accuracy in classifier fusion, Journal of Pattern Analysis and Applications 6 (2003) 22–31.
24. N. Meddouri, H. Khoufi, M. Maddouri, Diversity analysis on boosting nominal concepts, in: Pacific-Asia conference on Advances in Knowledge Discovery and Data Mining - Volume Part I, Springer, Berlin, Heidelberg, 2012, pp. 306–317.
25. P. Lenca, P. Meyer, B. Vaillant, S. Lallich, On selecting interestingness measures for association rules: User oriented description and multiple criteria decision aid, European Journal of Operational Research 184 (2) (2008) 610–626.